Vyprané partie z dátových štruktúr 2014: Domáca úloha
15% celkovej známky + nepovinné bonusové otázky
Termín odovzdania: štvrtok 3.4.2014 o 22:00

Cieľom tejto domácej úlohy je porovnať rýchlosť rôznych dátových štruktúr pre náhodné aj reálne dáta. Samotné dátové štruktúry implementujte v jazyku **C++ alebo Java**. Na spracovanie výsledkov, generovanie vstupov a pod. môžete použiť aj iné jazyky alebo programy.

Domácu úlohu odovzdávajte prostredníctvom systému Moodle, pričom odovzdajte dva súbory:

- Zozipovaný súbor obsahujúci zdrojový kód vašich programov (časti A, C a bonusové otázky). Dbajte na prehľadnosť a čitateľnosť programu a pripojte krátke README vysvetľujúce význam jednotlivých súborov a použitie vášho programu. Neodovzdávajte dátové súbory.
- Text, tabuľky, prípadne grafy v častiach B-D a v bonusových otázkach odovzdajte v jednom pdf súbore.

Neopisujte kód ani text od spolužiakov ani z internetu alebo iných zdrojov s výnimkou zdrojov povolených nižšie.

Časť A, 6 bodov: základná implementácia

Cieľ. Budeme uvažovať abstraktný dátový typ, ktorý udržuje množinu slov (reťazcov) a pre každé slovo počítadlo jeho výskytov, pričom umožňuje nasledujúce operácie:

- init(): vytvorí prázdnu štruktúru (táto operácia môže byť implementovaná ako konštruktor určitej triedy),
- add(w): zvýši počítadlo pre slovo w. Ak w ešte nie je v dátovej štruktúre, najskôr ho vloží,
- frequent(k): vráti (alebo priamo vypíše) všetky slová, ktoré sa vyskytujú aspoň k krát a počet výskytov každého z týchto slov.

Implementujte tri verzie tejto štruktúry:

- Nevyvažovaný binárny vyhľadávací strom, ktorý má ako kľúče slová a ako hodnoty počet výskytov. Ako pomôcku môžete použiť prednášku z prváckeho programovania: `http://compbio.fmph.uniba.sk/vyuka/prog/index.php/Prednáška_21`
- Splay strom. Použite pseudokód operácie splay uvedený v článku Sleator, Daniel Dominic, and Robert Endre Tarjan. "Self-adjusting binary search trees."Journal of the ACM (JACM) 32.3 (1985): 652-686 na strane 666 (pseudokód rotácie je na ďalšej strane). V každom vrchole si ukladajte smerník na rodiča. Operáciu splay(x) vykonajte po každej operácii add(w), pričom ju aplikujete na vrchol x, v ktorom je uložené slovo w.
- Vyhľadávací strom zo štandardných knižníc vášho jazyka (napr. map v knižnici STL pre C++ alebo TreeMap v Jave). Požadované funkcie implementujte ako vhodnú kombináciu metód poskytovaných knižničnou štruktúrou.

Operácia frequent(k) môže prehľadať celú dátovú štruktúru a vrátiť slová splňujúce podmienku, nemusíte sa teda snažiť udržiavať štruktúry umožňujúce rýchle nájdenie často sa vyskytujúcich slov.

Vstup. Váš program by mal na vstupe načítať postupnosť slov, jedno po druhom ich vložiť do dátovej štruktúry a na konci vypísať zoznam slov, ktoré sa vyskytovali aspoň k krát a počty ich výskytov. Vstupný súbor môže obsahovať ľubovoľný text. Znaky a-z a A-Z z anglickej abecedy zmeňte na malé písmená a považujte za súčasť slova. Ľubovoľné iné znaky (medzery, čísla, špeciálne znaky atď) považujte za oddeľovače slov.

Okrem hlavného vstupného súboru s textom by Váš program mal dostať (napríklad ako nastavenia na príkazovom riadku) prah k, ktorú implementáciu má použiť a ďalšie potrebné parametre.

Meranie času. Cieľom úlohy je porovnať rýchlosť uvedených troch implementácií. Budeme merať dve veličiny: samotný čas výpočtu (ten je ale ovplyvnený mnohými implementačnými detailami, kompilátorom, hardvérom a pod.) a počet porovnaní slov v dátovej štruktúre.

Ak váš program beží veľmi krátko, môže byť ťažké spoľahlivo odmerať čas výpočtu. Preto zopakujte celý algoritmus niekoľko krát a merajte celkový čas. Všetky experimenty spúšťajte na tom istom počítači a za podobných podmienok.

Počet porovnaní merajte tak, že si vytvorte pomocnú triedu, ktorá bude obsahovať samotné slovo (typu string) a v statickej premennej počítadlo porovaní. Táto trieda bude tiež implementovať lexikografické porovnávanie dvoch slov (zavolaním porovnávania pre stringy). Vždy keď sa toto porovnávanie spustí, zvýši sa počítadlo porovnaní. Pri vkladaní a hľadaní vo všetkých troch štruktúrach by ste na porovnávanie kľúčov v strome mali použiť práve toto lexikografické porovnávanie, najjednoduchšie je teda implmentovať ho ako metódu compareTo v Jave alebo operátor $<$ v C++.

Výstup. Výstupný súbor na prvom riadku obsahuje čas výpočtu vášho programu, počet porovaní a iné užitočné štatistiky v ľubovoľnom formáte, aký si zvolíte. Každý z ďalších riadkov má obsahovať jedno z výstupných slov malými písmenami, medzeru a počet výskytov tohto slova. Slová môžete vypísať v ľubovoľnom poradí.

Príklad. Uvažujme $k = 2$. Nasledujúci vstupný súbor obsahuje tri výskyty slova ma, dva výskyty slova mama a jeden výskyt slov ema, emu, mamu, sa. Jeden z možných správnych výsledkov je vpravo.

```
Vstup:                          Výstup:

Mama123ma Emu                   time: 1s comparisons: 14
Ema_*ma_mamu. Mama sa ma.       mama 2
                                ma 3
```

Časť B, 3 body

Na stránke predmetu nájdete dva súbory, každý obsahujúci prvých 20 000 slov z jednej knihy. Vo vašej úlohe pre každý z týchto súborov uveďte, ktoré slová sa vyskytujú aspoň $k = 250$ krát a počet ich výskytov. Tiež uveďte, koľko rôznych slov je v každom súbore ($k = 1$). Uveďte aj čas výpočtu a počet porovnaní pre všetky tri implementácie a pre každý súbor.

Časť C, 4 body

Napíšte program, ktorý vygeneruje z náhodných slov, každé dĺžky L nad abecedou veľkosti σ s nezávislými a rovnomerne rozdelenými znakmi. Uvažujte $z = 20000$, $L \in \{4, 10\}$ a $\sigma \in \{2, 26\}$. Pre každú z týchto štyroch kombinácií z, L a σ spustite všetky tri vaše implementácie na 10 náhodných vstupoch. Spočítajte priemerný čas výpočtu a smerodajnú odchylku (standard deviation) pre každú implementáciu a nastavenie parametrov. Výsledné štatistiky uveďte v prehľadnej tabuľke.

Časť D, 2 body

Okomentujte výsledné časy namerané v častiach B a C. Aké z nich vyvodzujete závery? Aké trendy pozorujete a čo ich spôsobuje? Správajú sa náhodné slová podobne ako prirodzený jazyk? Zodpovedajú namerané hodnoty vašim očakávaniam, alebo vás niečim prekvapili?

Nepovinné bonusové otázky

Za nasledujúce otázky môžete dostať bonusové body (tieto body sa nezapočítavajú do limitu 10% na bonusové body za aktivitu). Bonusové otázky budú bodované prísnejšie.

Časť E, 3 body

V splay strome sa navštívené slovo vždy presunie do koreňa a teda slová, ktoré sú na vstupe častejšie, by sa mali nachádzať bližšie ku koreni. Po načítaní celého súboru vypíšte na výstupe pre všetky slová s aspoň k výstupmi aj ich hĺbku v strome (toto správanie zapnite špeciálnym nastavením programu). Potom spustite program na vstupy z časti B a pokúste sa graficky alebo štatisticky vyhodnotiť, či vidíte závislosť medzi hĺbkou a počtom výskytov slova. Z tejto analýzy môžete vynechať slová s veľmi malým počtom výskytov. Porovnajte silu tejto závislosti v prípade splay stromov a nevyvažovaných stromov (kde by závislosť nemala byť príliš badateľná).

Časť F, 5 bodov

Spravte ďalšie experimenty na náhodných alebo reálnych dátach a pokúste sa objaviť nejaké zaujímavé trendy. Napríklad môžete experimentovať so zložitejšími modelmi náhodných slov, porovnávať rôzne prirodzené jazyky, preskúmať viac hodnôt parametrov L, z a σ, skúsiť meniť nastavenia kompilátora alebo vylepšiť čas malými zmenami v implementácii. Navrhnite vaše experimenty tak, aby odpovedali nejakú netriválnu otázku. Zdôvodnite návrh experimentov a rozdiskutujte namerané hodnoty. Pri bodovaní je dôležitejší vhodný návrh experimentov a vyvodené závery, než len počet experimentov, ktoré ste spravili.